SECTION 5: Bonus Challenge

Q9. Design an architecture to deploy an ensemble of AI trading agents that share a common alpha signal pool, execute asynchronously, and operate across multiple exchanges (Binance, Coinbase, CME) with low-latency requirements. Include:

● Execution layer

● Risk management

● Feature store

● Monitoring & logging

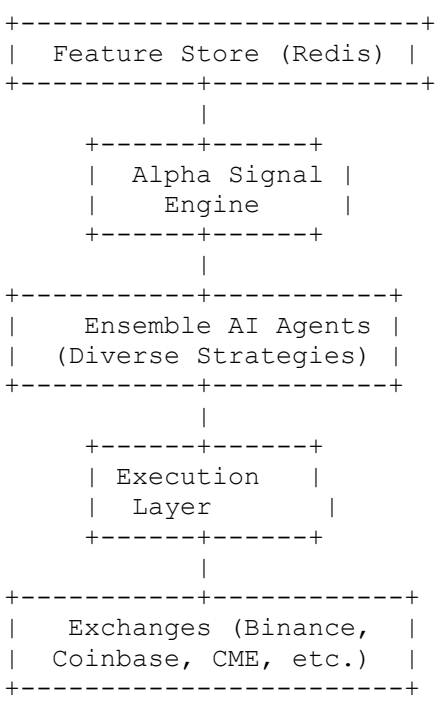
● Failover plan

Proof:

Here's a **simple and clear architecture** to deploy an **ensemble of AI trading agents** across multiple exchanges with low-latency execution:

Architecture Overview

Each part has a role:



1. Alpha Signal Pool

- A central module where real-time alpha signals are computed (e.g., from price patterns, order flow, indicators).
- Signals are shared with all AI agents.
- Stored in a fast in-memory store (like **Redis** or **Kafka**).

2. Ensemble of AI Agents

- Multiple agents trained on different strategies:
  - Trend-following, mean-reversion, arbitrage, etc.
- Each agent reads from the **shared alpha signal pool**.
- Agents **act independently** (asynchronously) based on their logic.
- Written in Python/C++ with async I/O to reduce latency.

3. Execution Layer

- A fast trading engine that:
  - Converts agent decisions into real orders.
  - Routes orders to the correct exchange (via API or FIX).
- Supports **multi-exchange order handling**.
- Uses **low-latency tech** (like ZeroMQ, WebSockets, or FIX protocol).

4. Risk Management

- Real-time limits:
  - Max position size
  - Max daily loss
  - Max exposure per exchange
- If a rule is broken → auto kill or reduce positions
- Managed by a **Risk Engine** that sits between agents and execution.

5. Feature Store

- Stores features used by AI models (e.g., volatility, volume, price momentum).
- Should be:
  - **Time-synchronized**
  - **Fast-access** (e.g., Redis, Apache Arrow, or Feather)
- Used for both training and real-time inference.

6. Monitoring & Logging

- Tools: **Prometheus + Grafana**, or ELK Stack (Elasticsearch, Logstash, Kibana)
- Monitors:
  - Latency
  - Trade slippage
  - Model predictions vs actual results
  - PnL, fill ratio, error rates
- Alerts for unusual patterns or drops in performance.

7. Failover Plan

- **Redundant agents**: If one fails, others continue.
- **Hot backup system**:
  - Real-time mirror of system that can take over in case of crash.
- **Health checks**: Ping services every few seconds.
- **Auto-restart scripts** using Docker, Kubernetes, or supervisor tools.
- **Persistent queues** (Kafka, RabbitMQ) to store unprocessed signals if crash happens.

Summary Table:

Component	Tool / Tech	Purpose
Alpha Signal Pool	Redis / Kafka	Share real-time trade signals
AI Agents	Python/Async, On Docker	Run strategies independently
Execution Layer	FIX, WebSocket, ZeroMQ	Fast order placement to exchanges
Risk Management	Custom engine + rules	Stop loss, exposure, max trade size
Feature Store	Redis / Arrow / Feather	Store and serve model features
Monitoring	Grafana + Prometheus / ELK	Watch system health and trading metrics
Failover Plan	Docker + Auto restart + Kafka queue	Keep system running in case of failure